UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRESSA CELANT

COMPARATIVO DE DESEMPENHO ENTRE MÉTODOS DE INDEXAÇÃO UTILIZADOS EM CONSULTAS DE DADOS ESPACIAIS

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2015

ANDRESSA CELANT

COMPARATIVO DE DESEMPENHO ENTRE MÉTODOS DE INDEXAÇÃO UTILIZADOS EM CONSULTAS DE DADOS ESPACIAIS

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Claudio Leones Bazzi.

Co-orientador: Prof. Nelson Miguel Betzek.

MEDIANEIRA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

COMPARATIVO DE DESEMPENHO ENTRE MÉTODOS DE INDEXAÇÃO UTILIZADOS EM CONSULTAS DE DADOS ESPACIAIS

Por

Andressa Celant

Este Trabalho de Diplomação (TD) foi apresentado às 13:00 h do dia 20 de novembro de 2015 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Analise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. Dr. Claudio Leones Bazzi
UTFPR – *Campus* Medianeira
(Orientador)

Prof. Valter Rodrigo Ekert
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Paulo Lopes de Menezes
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Me. Juliano Lamb
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

CELANT, Andressa. Comparativo de desempenho entre Métodos De Indexação utilizados em consultas de Dados Espaciais. 2015. Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

O trabalho tem como objetivo a realização de um estudo sobre possíveis problemas que o sistema gerenciador de banco de dados (SGDB) encontra para o armazenamento e consulta de dados espaciais. Além disso, será analisada as métricas resultantes das comparações entre métodos de indexação espacial, para encontrar possíveis melhorias de desempenho. Para tais testes de desempenho, foram realizadas cargas de trabalho em uma série de operações que resultaram em dados capazes de mensurar o desempenho de cada um dos índices aqui avaliados.

Palavras-chave: Dados espaciais, indexação espacial, banco de dados, B-Tree, Gist.

ABSTRACT

CELANT, Andressa. Performance comparison between De Indexing Methods used in queries Spatial Data. 2015. Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

The work aims to carry out a study on possible problems that Data Base Management System is for storing and querying spatial data. It also has the approach, analyze the metrics resulting from the comparisons between methods of spatial indexing and finding potential performance improvements. For such performance testing workloads was held in a series of transactions that resulted in data able to measure the performance of each of the indices evaluated here.

Keywords: spatial data, spatial indexing, database, B-Tree, Gist.

LISTA DE SIGLAS

BD	*Banco de Dados*
MIE	*Métodos de Indexação Espacial*
OMT-G	*Object Modelling Technique – Geographic*
ORM	*Modelo Objeto Relacional*
UML	*Unified Modeling Language*
SDBDOR	*Sistema Gerenciador de Banco de Dados Objeto Relacional*
SGBD	*Sistema Gerenciador de Banco de Dados*
SIG	*Sistema de Informação Geográfica*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento desse trabalho tem como fundamento a necessidade humana de definir e entender sua localização espacial, fenômenos naturais e socioeconômicos, ciência conhecida como Geografia. Tal disciplina depois do surgimento da informática, tem evoluído bastante, sendo que foi automatizada por meio do Geoprocessamento, que informatiza e processa dados georreferenciados. Essa automatização tem como principal ferramenta o Sistema de Informação Geográfica (SIG), que faz uso de uma estrutura de base de dados especial, podendo ser dividida entre dados geométricos, gráficos, espaciais ou atributos alfanuméricos e descritivos.

Por ter, o SIG, um campo muito vasto e versátil de manipulação, este estudo terá enfoque específico em banco de dados, ou seja, armazenamento e recuperação de dados espaciais. Este, também nominados de dados geográficos, em um SIG são basicamente diferenciados entre Vetorial e Matricial, uma vez que são representações da superfície terrestre composta por coordenadas geográficas. No caso de vetores nada mais são, por exemplo, que longitude e latitude, as quais são representadas por pontos, linhas ou polígonos, sendo X e Y de um vetor. Já no caso Matricial, ou Raster, essas coordenadas se tornam a coluna e linha de uma matriz, tendo cada célula dessa matriz um pixel com o seu valor representando uma grandeza física.

Todavia, há complexidade de compreensão dos dados espaciais no âmbito do banco de dados espacial, e por isso é recomendável fazer uso de algoritmos eficientes para a indexação, levando em consideração o tipo de dados a serem utilizados, a quantidade de dados por unidade de armazenamento e a estrutura de dados básica para obter o melhor desempenho. (GÜTING, 1994)

Portanto, esse trabalho tem como objetivo realizar uma análise sobre possíveis problemas de desempenho no banco de dados (BD) com uso de dados espaciais, bem como relatar resultados dos comparativos entre os Métodos de Indexação Espaciais (MIE).

1.1 OBJETIVO GERAL

Analisar, comparar e avaliar o desempenho entre métodos de indexação utilizados em consultas de dados espaciais.

1.2 OBJETIVOS ESPECÍFICOS

São objetivos específicos:

1. Estudar problemas de desempenho em consultas no banco de dados PostgreSQL fazendo uso de dados espaciais;
2. Comparar os principais índices espaciais, definindo vantagens, desvantagens e seu funcionamento;
3. Implementar algoritmos de indexação espacial;
4. Desenvolver uma avaliação entre os desempenhos dos métodos de indexação;

1.3 JUSTIFICATIVA

Banco de dados espaciais, como outros BD, devem garantir basicamente o armazenamento correto de seus dados e consultas eficientes.

No entanto, como se trata de dados com grande complexidade e volume, as consultas tendem a perder a sua eficiência, inclusive porque as buscas são realizadas com base em propriedades espaciais do objeto (LU; OO, 1993).

Para que tal fato não interfira na eficiência no processo de consultas no Sistema Gerenciador de Banco de Dados (SGDB), métodos de indexação são geralmente usados (ELMASRI ; NAVATHE, 2006). Há diversos MIEs, porém, ao utilizar esses métodos deve-se levar em consideração diversos fatores para que os mesmos realmente sejam uma ferramenta de auxílio, pois a eficiência dos índices varia de acordo a necessidade das operações e a estrutura do banco.

Por esta razão, dedica-se o estudo neste trabalho.

2 DESENVOLVIMENTO

2.1 SISTEMA DE INFORMAÇÃO GEOGRÁFICA (SIG)

O sistema de informação geográfica, ou geoprocessamento como é comumente conhecido, apesar de ser utilizado há muito tempo, começou a ganhar notoriedade a partir da década de 70, primeiramente pelo financiamento do estado e depois na década de 80, quando teve grande investimento por parte do setor privado e desenvolvimento de bases geográficas em grande escala continuamente até nos dias atuais.

O SIG, possui a finalidade de auxiliar a compreensão de informações geográficas, informações essas que segundo Ferreira (2006), podem se tratar tanto sobre locais na superfície da Terra, onde algo esta dada sua localização, ou ainda mais detalhadamente informações sobre, por exemplo, cada árvore em uma floresta, clima de uma grande região e densidade populacional de um país.

Informações geográficas podem ser volumosas ou estáticas. No caso da informação volumosa, pode, por exemplo, se tratar de um banco de dados que diariamente é inserido milhões de bytes de tipo de diferentes espécies de dados. As informações estáticas, essas por sua vez, podem ser informações já processadas e analisadas. Essas informações são compostas tanto de números, textos, fotogrametria, imagens de levantamento topográfico, geodésicos ou de satélites e radar, georreferenciamento e coleta de dados, listas, sons e mapas propriamente ditos.

A utilização de um Sistema Gerenciador de Banco de Dados para armazenar essas informações geográficas está substituindo a antiga forma que o SIG utilizava, onde os dados eram armazenados em arquivos internos. Uma vez que, a manipulação dos dados, como por exemplo, seleção e consulta, é feita através de uma interface entre o usuário e a máquina (CÂMERA; CASANOVA, 1996).

Esse auxílio do SIG, se da por meio de um processamento informatizado dos dados. Segundo Azemoy, Smith e Sicherman (1981), o SIG é uma ferramenta que fornece aos analistas da área, capacidades avançadas de armazenamento, acesso, manipulação e visualização de informação georreferenciada.

A multiplicidade de uso e conceitos desse tema requer definições consistentes sobre o mesmo, como por exemplo:

“Um sistema de suporte à decisão que integra dados referenciados espacialmente num ambiente de respostas a problemas” (COWEN, 1988).

“Um banco de dados indexados espacialmente, sobre o qual opera um conjunto de procedimentos para responder a consultas sobre entidades espaciais” (Smith, 1987).

“Ferramenta com avançadas capacidades de modelação geográfica” (Koshkariov, 1989).

Ou seja, compreende-se que o sistema é um conjunto de procedimentos, envolvendo softwares, hardwares e informações capazes de manipular características alfanumérica sem dados espaciais do mundo real.

2.2 DADOS ESPACIAIS

Esse estudo trata-se de uma classe particular dos dados espaciais, sendo ela a classe dos dados georreferenciados. Os dados espaciais abrangem qualquer tipo de dado que se refere alguma dimensão espacial ou geométrica.

Segundo Gatrell (1991), o espaço geográfico é entendido como um conjunto de objetos com relação definida entre eles. Os objetos tendo localização precisa na superfície da Terra e a relação entre eles sendo, por exemplo, a distância entre os mesmos. A localização pode ser dada através de um par de coordenadas em um plano Cartesiano em grandes escalas ou latitude e longitude em escalas menores.

Os objetos desse conjunto podem ter em comum características de informações de localização espacial (geométrica e topológicas), características de tempo e até mesmo características não-espaciais, descrevendo o objeto. Esses três componentes fundamentais são o que define um dado como geográfico (ARONOFF, 1991) .

Essa geo-ontologia para os dados espaciais, tem duas definições básicas sendo elas as realidades físicas e as sociais. Dentro de cada realidade mais uma divisão entre entidades individualizáveis e as com variação contínua.

A realidade física individualizável é aquela que se refere aos fenômenos físicos, ou seja, aos elementos do mundo natural, como por exemplo definições do terreno (cordilheiras, serras e vales). Já as entidades físicas com variações continuas, também chamadas de topológicas, associa-se a qualquer grandeza que varia continuamente (pluviosidade, estiagem e poluição) (CÂMARA, 2005).

O conceito social topográfico é um “conceito socialmente definido que ocorre no espaço de forma ininterrupto” e as entidades sociais individuais são aquelas que são regulamentadas, ou seja, tem um registro legal. (DPI.INPE.BR)

A relação definida entre os dados, em um SIG, pode representar três eixos de consulta dos mesmos: onde, o que e quando, além de combinações entre esses eixos. Em outras palavras, para a representação de um dado geográfico no computador tem que considerar a variação do tempo e do espaço.

2.2.1 Estrutura dos dados

Para Goodchild (1992), o SIG, é um modelo de representação dos dados espaciais, que se difere em relação às formas de percepção da realidade, podendo ser divididos entre a visão de objetos e a visão de campos, essas visões se referindo principalmente a estrutura dos dados espaciais.

A visão de objetos, demonstrada na Figura 1,é dada através do modelo vetorial, representa os elementos geográficos como pontos, linhas ou áreas e ainda uma mesma localização pode ser compartilhada com um ou mais objetos. Essa visão representa uma entidade que possui dimensões reais bem definidas, dada por um par de coordenadas absolutas e explicitas, sendo x e y (FRANK; GOODCHILD, 1990).

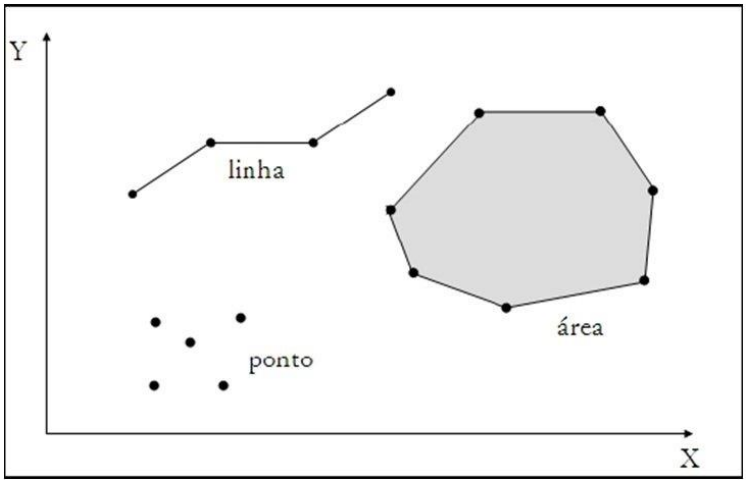


Figura 1 - Representações vetoriais em duas dimensões.
Fonte: DPI.INPE, (2004).

As duas visões se diferem principalmente na representação dos elementos, pois ao contrário do modelo de objetos, a visão de campos, representada pelo modelo raster, descreve

a realidade observada de uma superfície contínua. Essa representação é feita atrás de uma matriz regular composta por células quadradas, os chamados pixeis, organizados por linhas e colunas (FILHO; IOCHPE, 1996).

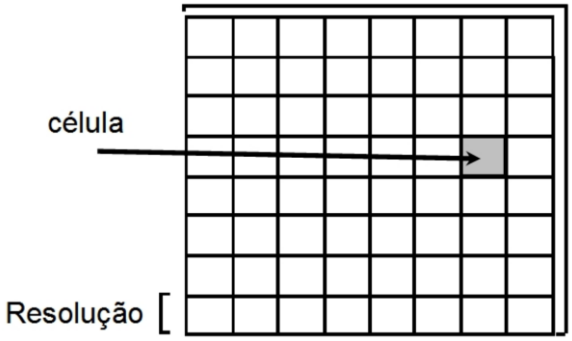
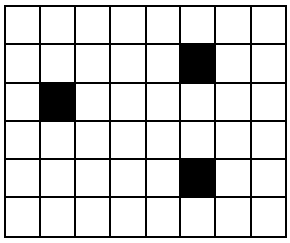
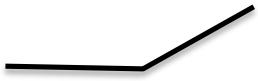
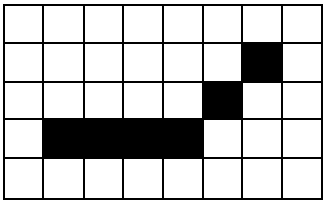
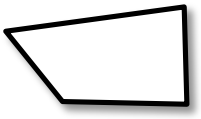
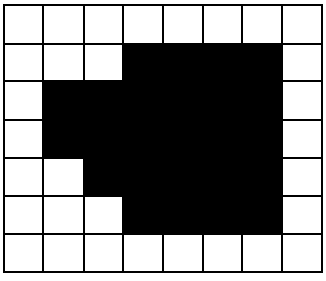


Figura 2 - Representação matricial dos dados geográficos.
Fonte: DPI.INPE, 2004

O Quadro 1 abaixo representa a comparação dos tipos de dados espaciais:

	Vetorial	Raster
Pontos		
Linhas		
Polígonos		

Quadro 1 - Tipos de dados espaciais
Fonte: Autoria própria

2.3 ARMAZENAMENTO (BANCOS DE DADOS)

Para o armazenamento é necessário primeiramente ter um modelo de dados geográficos que possibilita representar os diferentes tipos de dados espaciais e dados alfanuméricos. Essa representação engloba o suporte de relacionamentos, a representação gráfica e a manipulação dos dados obtidos de diferentes fontes (CÂMERA; BARBOSA, 2002).

A modelação de um banco de dados, parte do princípio de abstração das entidades do mundo real com o intuito de obter representações computacionais adequadas. O ato de abstrair e transformar em unidades lógicas de dados deve considerar o aspecto cognitivo na percepção espacial, a natureza dos dados (visão de campos ou objetos) e a existência de relações espaciais (BORGES; DAVIS, 2005).

A integração entre os dados espaciais e os convencionais é fundamental, pois dados espaciais são formados por tabelas de atributos. Essas tabelas são compostas por colunas que indicam as coordenadas geográficas e relacionada a elas um conjunto de atributos descritivos sobre essa coordenada, como por exemplo, atributos dos arcos quando o dado armazenado é um polígono ou até mesmo informações sobre os relacionamentos dados armazenados em linhas (FILHO, 1995).

Modelo de dados OMT-G é o principal modelo que atende essas necessidades, pois prove tanto a parte de representação dos dados geográficos quanto dos alfanuméricos, partindo das premissas de um diagrama de classes da Unified Modeling Language (UML). Devido ao uso de pictogramas representando a geometria dos objetos, esse esquema é mais compacto, intuitivo e de fácil compreensão (BORGES; DAVIS, 2005).

O diagrama de classes do modelo, pode conter classes georreferenciada ou convencionais, permitindo visualizar a estrutura do BD juntamente com seus relacionamentos, podendo assim derivar um conjunto de restrições de integridade espaciais (condições que precisam ser garantidas para que o banco de dados seja íntegro).

O conceito de Classes Georreferenciadas são classes que se especializam em dados do tipo geo-campo, onde representam objetos distribuídos continuamente pelo espaço e geo-objeto, que representam objetos geográficos individualizáveis, que possuem identificação com elementos do mundo real (BORGES, 2005).

A partir desse conceito de Object Modelling Technique - Geographic (OMT-G) é possível realizar o mapeamento dos objetos e classes para a criação de um banco de dados,

conforme demonstra a Tabela 1. Geralmente sendo atendido através do Modelo Objeto-Relacional (ORM), onde as classes do modelo conceitual OMT-G serão as tabelas.

O Modelo Objeto-Relacional, que integra as funcionalidades do Modelo Relacional com o da Orientação a Objetos, permite a definição de Tipos Abstratos de Dados e a Manipulação de Objetos Complexos.

Tabela 1 - Representações dos dados geográficos em OpenGis

Representação OMT-G		*Representação OpenGIS (Simple Features Specification)*
Geo-objeto	Ponto	Point
Geo-objeto	Linha	LineString
Geo-objeto	Polígono	Polygon
Geo-objeto	Nó de rede	Point
Geo-objeto	Arco unidirecionado	LineString
Geo-objeto	Arco bidirecionado	LineString
Geo-campo	Amostras	Point
Geo-campo	Isolinhas	LineString e/ou Polygon
Geo-campo	Subdivisão planar	Polygon
Geo-campo	Triangulação	Point (vértices) e Polygon (triângulos)
Geo-campo	Tesselação	GeoRaster, campo binário longo

Fonte: (BORGES, DAVIS e LAENDER, 2005)

2.4 POSTGRESQL E POSTGIS

PostGreSQL é um sistema de gerência de banco de dados objeto-relacional, gratuito e de código fonte aberto. Foi desenvolvido a partir do projeto Postgres, iniciado em 1986, na Universidade da Califórnia em Berkeley. No caso de dados espaciais, existe extensão para modelagem física, baseadas nas especificações do OpenGIS, o PostGIS (POSTGRESQL, 2015).

O PostGIS inclui suporte a índices espaciais baseado em GIST R-Tree, e funções para análise e processamento de objetos SIG. Segundo o PostGIS (2015), os tipos de dados que ele manipula são:

- *Point* - um único local, com x e y (longitude e latitude). Ex.: Uma planta.
- *Linestring* - sequência de pontos e os segmentos de linha que os conectam. Ex.: Uma curva de nível.
- *Polygon* - superfície bidimensional armazenada como uma sequência de pontos. Para ser considerado um polígono, o ponto inicial deve ser o mesmo do ponto final. Ex.: contorno de uma área territorial.
- *Multipoint* - uma coleção de zero ou mais pontos. Ex.: uma plantação de soja, onde cada ponto é uma planta de soja.
- *Multilinestring* – um conjunto de linhas. Ex.: diversas curvas de níveis de uma plantação.
- *Multipolygon* - conjunto de polígonos. Ex.: varias áreas de plantio de um município.
- *Geometrycollection* – coleção de todas as formas geométricas. Ex.: Um município com diversas áreas de plantio, com suas devidas curvas de níveis e suas plantas.

2.4.1 Consultas espaciais

Consultas espaciais, nada mais são do que comandos SELECTs de banco de dados convencionais com uma combinação de SQL que retornam valores específicos ou funções, as quais podem fazer parte de um índice (POSTGIS, 2015).

Além disso o PostGIS oferece diversos tipos de operadores espaciais divididos em operadores topológicos, operador de construção de mapas de distância, operador para construção do fecho convexo, operadores de conjunto, operadores métricos, centroide de geometrias e validação.

A Tabela 2, classifica os tipos de operadores com as respectivas funções que podem ser aplicadas.

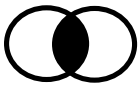
Tabela 2 - Operadores de dados geográficos

Tipo de Operador	Funções
Operadores topológicos	*equals(geometry, geometry)* *disjoint(geometry, geometry)* *intersects(geometry, geometry)* *touches(geometry, geometry)* *crosses(geometry, geometry)* *within(geometry, geometry)* *overlaps(geometry, geometry)* *contains(geometry, geometry)* *relate(geometry, geometry)*
Operadores de conjunto	*Intersection(geometry, geometry)* *geomUnion(geometry, geometry)* *symdifference(geometry, geometry)* *difference(geometry, geometry)*
Operadores métricos	*distance(geometry, geometry)* *area(geometry)*
Operador de construção de mapas de distância	*buffer(geometry, double, [integer])*
Operador para construção do Fecho Convexo	*Convexhull(geometry)*
Centróide de geometrias	*Centroid(geometry)*
Validação	*isSimple(geometry)*

Fonte: Autoria própria

Para melhor compreensão de como utilizar esses operadores é necessário entender os três principais tipos de consultas que são as consultas de proximidade, consultas de região e de consultas de interseção ou união, segundo (SILBERSCHATZ; KORTH, 1999).

O Quadro 2, exemplifica cada um dos tipos de consultas.

Tipo da Consulta	Exemplo / Figura
Interseção ou União	Dado dois conjuntos: A união de A e B será composto pelos elementos que pertencem, pelo menos, a um dos conjuntos. (A ∪ B)  A interseção de A e B será composto pelos elementos que pertencem aos conjuntos A e B simultaneamente.(A ∩ B) 
Região	Esse tipo de consulta trabalha com regiões espaciais procurando por objetos que estão inseridos, inteiramente ou parcialmente, em outra região. Onde o indicador "JC" na figura abaixo é a região consultada. 
Proximidade	Geralmente buscam por atributos que se localizam a certa distância de um ponto referência. Na figura abaixo o retângulo demonstra a área que esta sendo consultada e pontos pretos os atributos próximos à essa área. 

Quadro 2 - Tipos de consultas espaciais

Fonte: Autoria própria

2.4.2 Comando Explain

O comando *explain* do PostgreSQL é uma visão interna do SGBD que torna possível a análise de desempenho de uma operação, tanto de consulta, inserção, exclusão até mesmo a operação *execute* e *declare* de funções do banco. Dessa forma auxilia na tomada de decisão na hora de melhorar a performance do banco de dados (OLIVEIRA, 2007).

Fonte: (OLIVEIRA, 2007)

```
EXPLAIN [ ANALYZE ] [ VERBOSE ] statement
```

Quadro 3 - Comando explain

Esse comando demonstra como foi percorrido as tabelas para a execução das operações resultando uma série de informações, como por exemplo se os dados estavam ordenados, se há várias tabelas referências, entre outros. Mas o mais importante dele para a análise de desempenho é o resultado de custo estimado de execução do comando, ou seja, a estimativa de tempo que levará para executar certa operação. Há o modo de visualizar o tempo real e não somente o estimado através da opção *analyse*, a qual realmente executa o *statement* (operação) (RICARDO; PIRES, 2013).

O retorno do *explain* alguns códigos que precisam ser entendidos para um bom aproveitamento:

- *Seq Scan on curso*:
- *Cost*: dividido entre tempo de inicialização e o tempo máximo de execução
- *Rows*: número de linhas retornadas
- *Widht*: tamanho médio (em *bytes*) das linhas retornadas.

A Figura 3, é um exemplo de uma consulta executada através do comando *explain.*

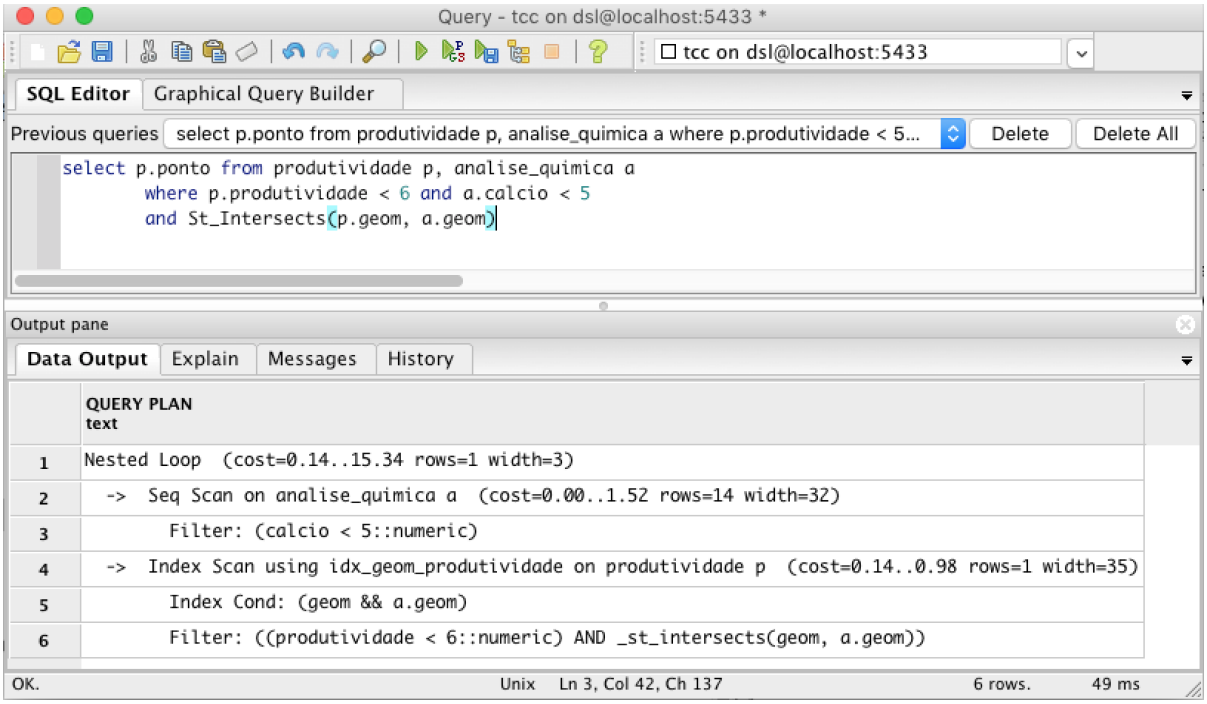


Figura 3 - Exemplo de retorno do método explain
Fonte: Autoria própria

2.5 ÍNDICES

Um índice é uma estrutura organizada de dados derivados de uma tabela, estrutura essa que por sua vez permite acesso aleatório dos dados o que reduz os testes computacionais, dessa forma deixando de ser necessária a busca linha a linha dos registros de uma tabela, que é a forma padrão de consultas dos SGBD (MAGUIRE; LONGLEY, 2012).

Utilizados para elevar a performance dos bancos de dados, os índices devem ser implementados com sensatez, pois produzem um trabalho adicional para o sistema de banco de dados podendo deixa-lo lento ao invés de elevar sua performance de busca (POSTGRESQL, 2015). Segundo Bianchi (2007), deve-se considerar alguns fatores importantes antes da criação como sua manutenção, pois consome tempo, dessa forma acaba consumindo recursos, não utilizar em colunas com pouca variação e sim em colunas com grande seletividade.

A própria ferramenta PostgreSQL disponibiliza diversos tipos de índices, além de ser possível definir métodos próprios, entretanto, o presente trabalho visa realizar um estudo sobre dados espaciais, analisando tipos de índices conhecidos (B-Tree e GIST) e verificar se há

melhora no desempenho na execução de consultas envolvendo operações espaciais entre os mesmos.

A sua criação básica é relativamente simples que pode ser vista no Quadro 4:

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] nome_do_índice ON
nome_da_tabela [ USING nome_do_método ]
    ( { coluna | ( expressão ) }
    [ classe_de_operadores ] [, ...] )
    [ WITH (parâmetro_de_armazenamento = valor [, ...] )]
    [ TABLESPACE espaço_de_tabelas ]
    [ WHERE predicado ]
```

Quadro 4 - Criação de um índice
Fonte: (POSTGRESQL, 2015)

Onde o "*nome_do_indice"* é como o mesmo será chamado na tabela (*"nome_da_tabela*") que foi especificada, podendo ser qualificada pelo seu "esquema" utilizando o método citado, por padrão a ferramenta PostgreSQL utiliza o método B-Tree.

Os campos chave para o índice são especificados como nomes de coluna ou, também, como expressões escritas entre parênteses. Podem ser especificados vários campos, se o método de índice suportar índices multi-colunas.

O *TABLESPACE* específica onde o índice será criado, por padrão é utilizado o *default_tablespace.*

As restrições dos índices são determinadas pelo seu predicado, ou seja, sua cláusula *where*.

Os parâmetros *UNIQUE* e *CONCURRETLY* correspondem a gerar uma exceção quando uma tentativa de inserir ou atualizar dados já existente e o parâmetro *CONCURRETLY* não bloqueia o banco de dados na construção do índice, ou seja, permite inserções, atualizações e exclusões da tabela onde o mesmo esta sendo criado (OLIVEIRA, 2007).

Há a possibilidade de alteração dos índices, podendo renomear o mesmo, mudar seu *tablespace*, alterar ou redefinir um ou mais parâmetros de armazenamento específico do método do índice (OLIVEIRA, 2007). Alterações realizadas através dos comandos demonstrados abaixo no Quadro 5:

```
ALTER INDEX nome_do_indice RENAME TO novo_nome

ALTER INDEX nome_do_indice SET TABLESPACE
   nome_do_espaço_de_tabelas

ALTER INDEX nome_do_indice SET (parâmetro_de_armazenamento
   = valor [, ... ] )

ALTER INDEX nome_do_indice RESET
   (parâmetro_de_armazenamento [, ... ] )
```

Quadro 5 - Alteração de um índice
Fonte: (POSTGRESQL, 2015)

O comando "*drop index*" também existe e podendo ser executado definindo alguns parâmetros. O "*cascade*" funciona removendo objetos que são dependentes do índice a ser excluído, já no caso do "*restrict*" a remoção do índice é bloqueada se houver algum objeto dependente, no caso, esse é o padrão. Por fim, o "*if exists*" a ferramenta lança uma exceção se o índice definido pelo "*nome_do_indice*" não existir no banco (OLIVEIRA, 2007). Exemplificado no Quadro 6:

```
DROP INDEX [ IF EXISTS ] nome [, ...] [ CASCADE | RESTRICT ]
```

Quadro 6 - Remoção de um índice
Fonte: (POSTGRESQL, 2015)

2.5.1 B-Tree (Árvore balanceada)

Em 1979, a indústria de *software* depois de um longo período conseguiu resolver o problema para bases de dados que não paravam de crescer. Foi aplicando uma fórmula matemática algorítmica criada em 1972, por alguns cientistas matemáticos de uma renomada universidade. Essa fórmula algorítmica é chamada de B-Tree, ou árvore balanceada, que cria uma lógica diferente para o uso de índices, de forma a tornar as pesquisas muito rápidas em bancos de dados gigantescos (SANTANA, 2012).

A Figura 4, é um exemplo de organização dos nós em uma árvore balanceada.

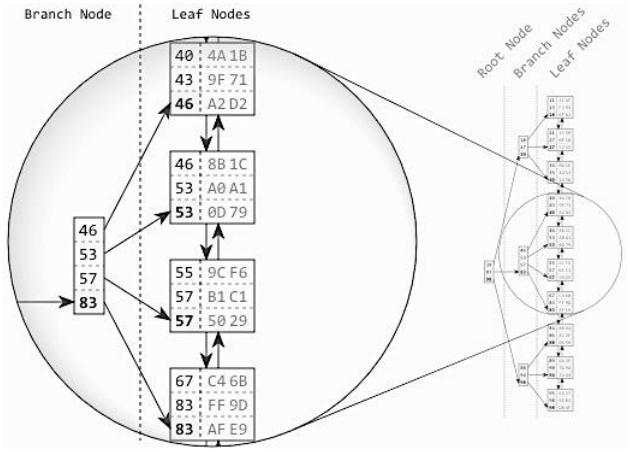


Figura 4 - Índice B-Tree
Fonte: (WINAND, 2011)

A indexação baseada em árvore organiza a entrada de dados de maneira ordenada pela forma hierárquica. A letra "B" do nome do índice B-Tree quer dizer *balanced*, o qual significa que se trata de uma árvore com todos os lados parecidos, ou seja, gerencia o espaço usado por seus blocos para que ele sempre esteja ocupado com pelo menos a metade de sua capacidade. E o principal fator que faz B-Tree se diferenciar dos outros é porque apresentam sua altura mais reduzida que as demais árvores de busca, podendo ser usadas para implementar operações com tempo computacional O(log n), que é considerado um tempo excelente para estas operações numa grande quantidade de arquivos armazenados. Praticamente todos os bancos de dados SQL baseiam-se nesse padrão de buscas.

A busca na B-Tree é similar a uma busca em uma árvore binária, exceto pelo fato que ao invés de apenas fazer uma decisão binária para qual ramo da árvore prosseguir a cada nó, a decisão passa a ser em relação ao número de chaves que o nó possui. Ou seja, no pior caso, é feito n[x] + 1 decisões, sendo n[x] o número de chave em um nó.

O número de acessos a disco realizados pela busca é da ordem da altura h da árvore, portanto, O(h) = O(logt n) onde *n* é número de chaves na árvore e *t* é o mínimo de chaves por nó. Lembrando que cada nó comporta no máximo *2t* chaves, o tempo computacional de busca da B-Tree é dado por O(th) = O(tlogt n) (PEDER; SCHUCK, 2009).

Geralmente índices do tipo B-Tree, são utilizados para consultas de igualdade e de faixa que possuem dados que podem ser ordenados segundo algum atributo. Nesse caso, por ser unidimensional não é recomendado para indexação de dados espaciais (MAGUIRE; LONGLEY, 2012).

O gerenciador leva em consideração utilizar um índice B-Tree sempre que uma coluna indexada estiver envolvida com um dos operadores: menor (<), menor ou igual a que (<=), igual (= ou *like*), maior ou igual a que (>=) e maior (>) (Chapter 11. Indexes, 2014).

Para os operadores LIKE e ILIKE, o B+Tree só deve ser usado quando se deseja buscar um padrão constante de caracteres encontrados apenas no início da cadeia. Dessa forma, consultas do tipo: LIKE ABCDE% são permitidas e, não são permitidas consultas do tipo: LIKE %ABCDE%.

Operadores como *between* e *in* também podem ser implementados, porém deve ser pouco utilizados, uma vez que algumas consultas podem comparar com valores nulos e que não são indexáveis, o que pode tornar a consulta lenta (POSTGRESQL, 2015).

2.5.2 R-Tree

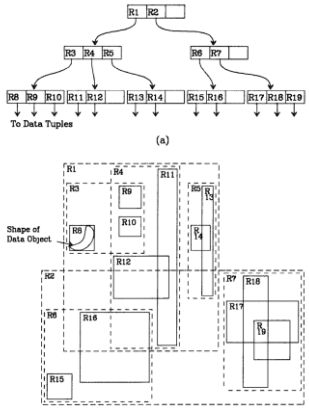


Figura 5 - Representação R-Tree
Fonte: (TOMA, 2008).

O método R-Tree foi criado para trabalhar com dados multidimensionais, ou seja, dados mais complexos, tais como geográficos, já bem consolidados, espaço-temporais e também dados de multimídia, como por exemplo imagens, voz, música ou vídeos, que ainda estão sendo aperfeiçoados. Dessa forma, tornando altamente indicado para tratamentos de consultas com dados espaciais (TOMA, 2008).

Como o R-Tree foi originalmente proposto por Guttman era baseado no método B-Tree, suas características são semelhantes, como por exemplo são árvores de altura equilibradas. Porém como o objetivo era a eficiência de grandes coleções de dados multidimensionais, o

método R-Tree organiza os seus objetos por meio de uma hierarquia de retângulos (chamados de MBRs - Minimum Bounding Rectangle) para o acesso dinâmico (GUTTMAN, 1984).

O PostgreSQL nas versões anteriores a 9.4 considera utilizar esse método quando a coluna que esta sendo indexada utiliza algum dos operadores de comparação como: << (está à esquerda), &< (não está à direita), &> (não está à esquerda), >> (está à direita), <<| (vertical a esquerda), |>> (vertical a direita), ~ (contém), @ (está contido sobre), ~=(o mesmo que) (TYPES, 2011).

Entretanto, após a versão do PostgreSQL 9.4, este método foi removido porque não tinha vantagens significativas em relação ao método GIST.

2.5.3 GIST (Generalized Search Tree)

GIST empacota os principais métodos de busca em árvore (há uma longa lista que inclui R-Tree, B+-Trees, HB-árvores, árvores de TV, Ch-Árvores, soma parcial e muitos outros), tornando a implementação de consultas de diversos tipos de dados descomplicada. Além de unificar todas essas estruturas, o GIST tem uma característica chave que faltava árvores anteriores: os dados e extensibilidade consulta, ou seja, é possível um índice de método B-Tree aceitar qualquer tipo de dado, mas com o mesmo principio de operadores. Em suma, GIST combina extensibilidade juntamente com generalidade, a reutilização de código, e uma interface limpa (HELLERSTEIN, 1999).

Como é um método que foi baseado no B-Tree, o GIST também é uma árvore equilibrada, mas diferente do que no B-Tree onde a chave é um inteiro, nesse caso são classes, onde deve se identificar qual o conteúdo da mesma e implementar 4 métodos para a classe chave que ajudam a árvore fazer a inserção, exclusão e pesquisa (HELLERSTEIN, 1999).

No PostgreSQL são oito métodos possíveis de implementação, sendo eles: *consistente*, *union*, *compress*, *decompress*, *penality*, *picksplit*, *same*, *distance* (POSTGRESQL, 2014).

Desses oito, quatro deles são fundamentais:

- *Consistent*: Este método permite a busca árvore corretamente. Dada a chave (p) em uma página da árvore, e a consulta (q), o método consistente deve retornar NO se é certo que ambos p e q não podem ser verdade para um determinado item de dados. Caso contrário, ele deve retornar *MAYBE* (HELLERSTEIN, 1999).

- *Union*: Este método consolida informação na árvore. Dado um conjunto de entradas, este método retorna uma nova chave que é verdade para todos os itens de dados abaixo da entrada (HELLERSTEIN, 1999).
- *Penality*: Retorna um valor que indica o "custo" de inserir a nova entrada em um determinado ramo da árvore. Os itens serão inseridos para baixo o caminho de menor onerosidade na árvore. Valores devolvidos com muito custo deve ser não-negativo. Se um valor negativo for devolvido, ele será tratado como zero. Essa função é muito importante para o bom desempenho do índice, pois como trata já da inserção do dado, no momento da pesquisa, com os já organizados, se tornara mais rápida (POSTGRESQL, 2014).
- *Picksplit*: Como numa B-tree que as entradas devem ser divididas para tornar a arvore mais equilibrada, essa rotina é responsável por separar as páginas, decidindo em qual deve se posicionar os dados.

O GIST vem substituindo o indice R-Tree essencialmente por dois fatores. Um deles é que utilizando o GIST é possível controlar objetos maiores que 8K, o que não é possível no R-Tree e além disso o GIST é um índice do tipo "*null-safe*", que significa que é seguro em buscas que contenham dados nulos, não falharam (POSTGIS, 2015).

Segue abaixo Quadro 7, que apresenta os operadores de classes suportados na distribuição PostgreSQL que incluem o GIST:

Nome	**Tipo de dados**	**Operadores**
box_ops	Box	&&, &> , &<, &<|, >>, <<, <<| ,<@, @>, @ |&>, |>>, ~, ~=
circle_ops	Circle	&&, &>, &< ,&<|, >> ,<< ,<<|, <@ ,@>, @|&> ,|>>, ~ ,~=
inet_ops	Inet, cidr	&&, >>, >>=, >, >=, <> ,<< ,<<= ,<, <=, =
point_ops	Point	>>, >^, << <@ ,<@ ,<@ <^, ~=
poly_ops	Polygon	&&, &>, &< &<|, >> << , <<|, <@, @> ,@|&>, |>>, ~, ~=
range_ops	Vários tipos de dados	&&, &>, &< ,>> ,<< ,<@, -|-, = ,@> @>
tsquery_ops	Tsquery	<@ @>
tsvector_ops	tsvector	@@

Quadro 7 - Operadores de classes GIST

Fonte: (POSTGIS, 2015)

2.5.4 Índice Quadtree

O método Quadtree é uma estrutura hierárquica com o principio da decomposição recursiva de espaços. A decomposição pode ser feita em partes iguais em cada nível (decomposição regular), ou pode depender dos dados de entrada. O espaço de busca é recursivamente decomposto em quadrantes até que o número de retângulos sobrepondo cada quadrante é menor do que a capacidade da página. Em duas dimensões cada ponto é representado por um quadrante que possui quatro ou nenhum filho (NONATO; VOORSLUYS, 2000).

Cada quadrante de um nó representa o NW (nordeste), NE (noroeste), SW (sudoeste), SE (sudeste) do conjunto de dados. (KOTHURI; RAVADA, 2002)

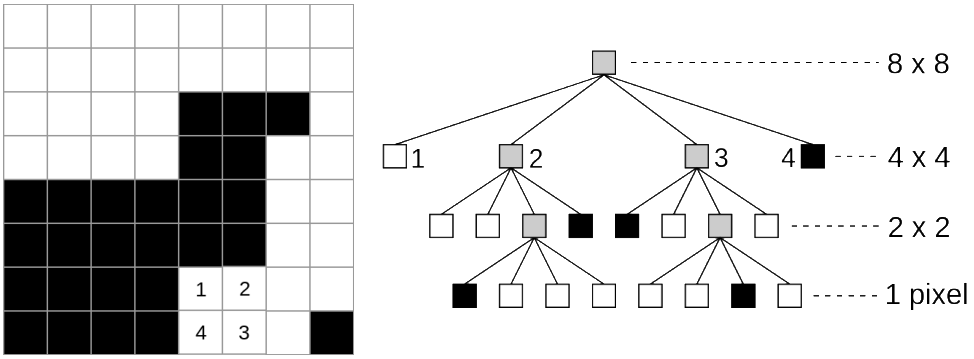


Figura 6 - Demonstração gráfica Quadtree
Fonte: (Quadtree, 2015).

2.5.5 Kd-Tree

Kd-Tree é um método de busca em árvore binária de varias dimensões, geralmente utilizados em consultas de proximidade de um ponto por se caracterizar eficiente, onde o resultado esperado é um conjunto de pontos contidos nesse raio.

Há dois tipos de Kd-Tree, o *Original* e o *Adaptive*, sua diferenciação se da pelo modo de armazenamento do dado em cada nó da árvore. No tipo *Original* o método de inserção pode resultar em uma árvore não equilibrada, pois a cada entrada é analisado com base nas suas coordenadas em que nó o novo ponto ira pertencer. Já no *Adaptive*, como os valores dos dados

são somente armazenados nas folhas da árvore, é possível criar uma estrutura equilibrada, pois os nós intermediários armazenam apenas informações de construção e para o encaminhamento correto da arvore. Dessa forma, as consultas ficam ainda mais eficientes pois o número de nós percorridos é menor devido ao caminhamento. Porém, para a construção de uma árvore do tipo *Kd-Tree Adaptive* é necessário que se tenha todos os dados que serão armazenados previamente, para depois aplicar o método de indexação (DAVIS JR.; QUEIROZ, 2005).

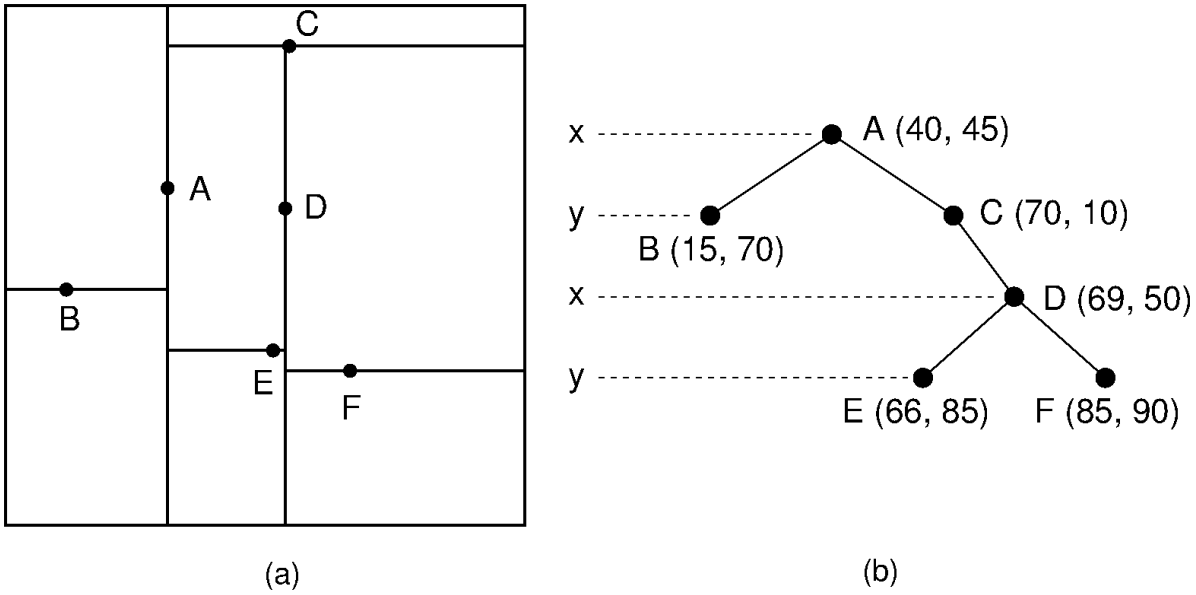


Figura 7 - Representação gráfica Kd-Tree Original
Fonte: (YE, 2014)

3 MATERIAIS E MÉTODOS

3.1 FERRAMENTAS UTILIZADAS

No trabalho foi utilizado os *softwares* PostgreSQL na sua versão 9.4, gratuita fazendo uso da extensão PostGis, na versão 2.1, através da interface gráfica PgAdmin III.

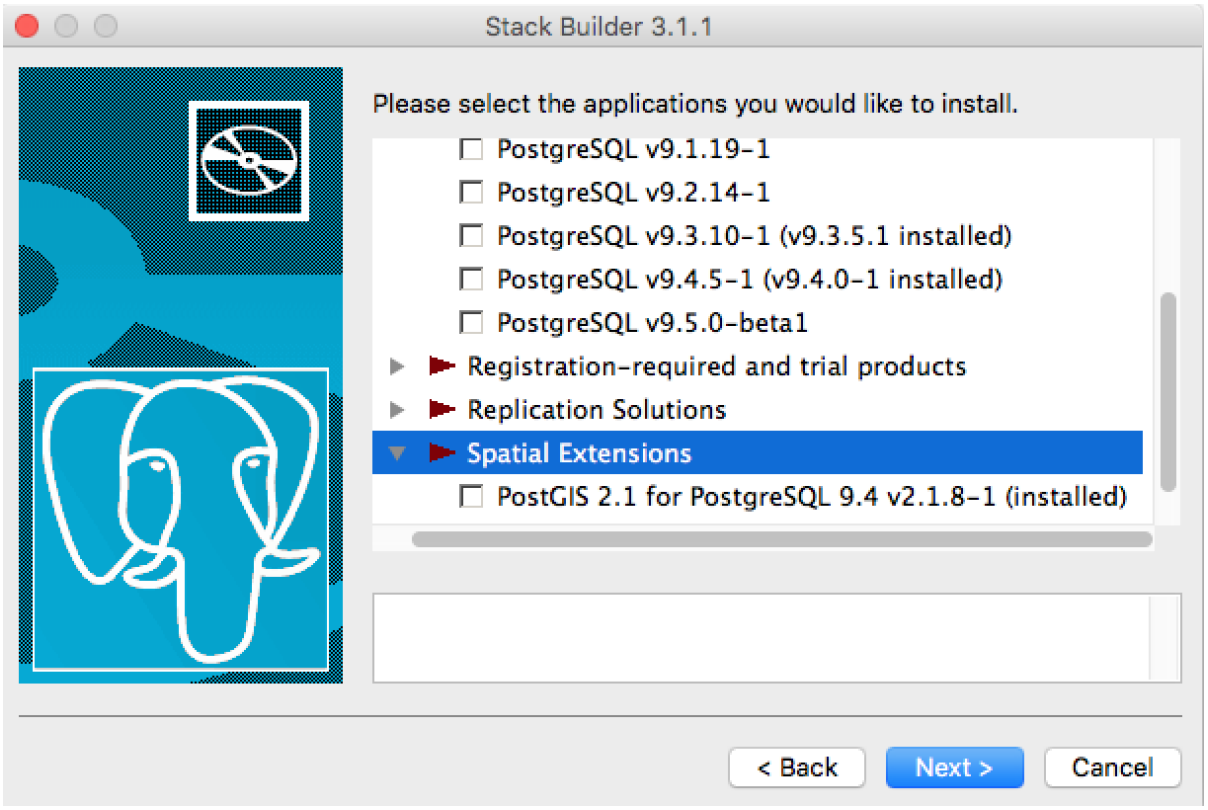


Figura 8 - Versão do SGBD utilizada
Fonte: Autoria própria

Esses softwares instalados em uma máquina composta por processador Intel® Core™ i5-2310 CPU @ 2,6 GHz com um número total de núcleos igual a 2 e cada núcleo composto de cache L2 de 256KB. A memória RAM de 8GB of 1600MHz DDR3L onboard memory; HD APPLE SSD SD0128F (6 NAND Flash 05131 016G de 16GB, totalizando 128GB de capacidade de armazenamento) para sistema operacional e arquivos do banco de dados.

3.2 DADOS UTILIZADOS

Os dados utilizados para a geração das análises sobre dados do tipo *point* foram coletados em uma área agrícola, cultivada sob sistema de plantio direto com sucessão de culturas soja e milho há pelo menos 10 anos. A área está localizada no município de Serranópolis do Iguaçu/Paraná, sob coordenadas geográficas centrais 25º24'28" S e 54º00'17" O, com elevação média de 355 m. Dados esses referentes a produtividade que foi obtida na nessa área na safra de soja no final do ano 2012 e começo de 2013, além da análise química do solo realizada antes do plantio.

Para dados do tipo *Multipoint*, *MultiLineString*, *MultiPolygon*, foi utilizado a Base Cartográfica Contínua do Brasil na escala 1:250.000, que reúne dados geoespaciais de referência, estruturados em bases de dados digitais, permitindo uma visão integrada do território nacional nesta escala. Estão contempladas as seguintes categorias de informação: Hidrografia, Sistema de Transporte, Energia e Comunicações, Abastecimento de Água e Saneamento Básico, Estrutura Econômica, Localidades e Limites.

3.3 FLUXO DO DESENVOLVIMENTO

A Figura 9 exemplifica como se deu o desenvolvimento dessa análise:

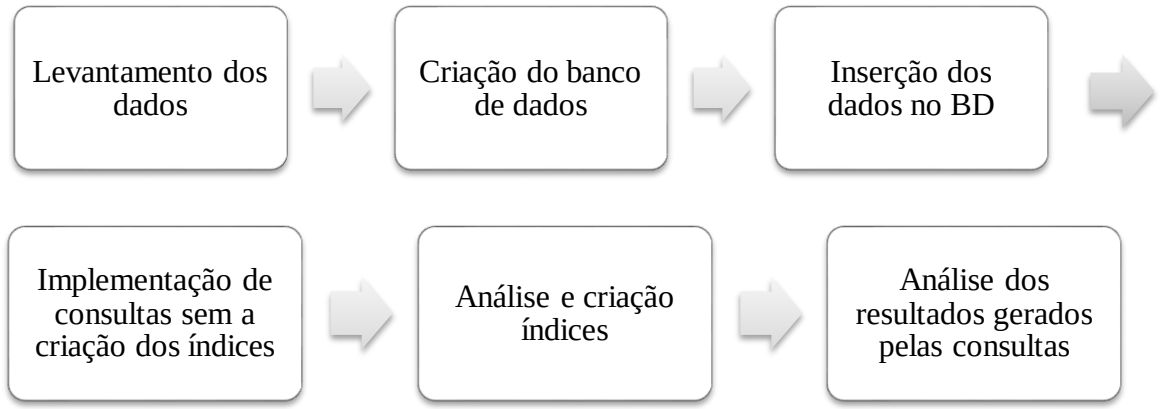


Figura 9 - Fluxo de desenvolvimento da análise
Fonte: Autoria própria

3.4 BANCO DE DADOS

A estrutura do banco de dados do estudo, contém duas tabelas que representam: uma os dados sobre a produtividade e a outra a análise química do solo, ambos os dados são referentes a área de Serranópolis descrita anteriormente, sendo que as tabelas não têm relacionamento entre si.

Para cada tabela existe um atributo do tipo *geometry*, onde é armazenado os dados de localização, sendo estes, longitude e latitude, que são do tipo *point*. São exemplo de criação e inserção da tabela de produtividade respectivamente:

- CREATE TABLE produtividade(
  id bigint NOT NULL DEFAULT nextval('produtividade_a_id_seq'::regclass),
  ponto character varying(4),
  produtividade numeric,
  geom geometry,
  CONSTRAINT pk_produtividade_id PRIMARY KEY (id));
- INSERT INTO produtividade(ponto, produtividade, geom)
  VALUES ('A1', 4.509, ST_GeomFromText('POINT(-54.006368 -25.408065)', 4326)));

Além disso, há no mesmo banco tabela referente a um conjunto de elementos agregados envolvendo componentes de um sistema de saneamento básico sendo dados referentes aos cemitérios brasileiros com dados do tipo *multipoint* e também uma tabela que envolve componentes do sistema industrial. Na Figura 10, representação da criação da tabela com tipo de dado *multipoint*.

```
CREATE TABLE asb_cemiterio_p (
    id_objeto integer NOT NULL,
    nomeabrev character varying(50),
    geometriaaproximada character varying(3),
    denominacaoassociada character varying(30),
    tipocemiterio character varying(30),
    nome character varying(80),
    geom public.geometry(MultiPoint,4674) NOT NULL
);
```

Figura 10 - Criação da tabela com dados multipoint
Fonte: Autoria própria.

Os dados do tipo *multipolygon*, estão contidos em uma tabela que compreende um conjunto de elementos agregados envolvendo componentes que abrangem a exploração ordenada dos recursos naturais, vegetais e animais em ambiente natural e em ambiente protegido. Tabela demonstrada na Figura 11.

```
CREATE TABLE eco_area_agropec_ext_vegetal_pesca_a (
    id_objeto integer NOT NULL,
    geometriaaproximada character varying(3),
    destinadoa character varying(30),
    geom geometry(MultiPolygon,4674) NOT NULL
);
```

Figura 11 - Criação de tabela com tipo de dado Multipolygon
Fonte: Autoria própria.

Além da tabela "eco_area_agro_ext_vegetal_pesca_a", foi inserido mais uma tabela para aplicação da função St_Intersects no banco ao se tratar do tipo de dados *multipolygon*, a qual trata todas as Unidades da Federação em forma de polígonos.

A tabela "enc_treco_energia_l" compõe um conjunto de dados que descrevem linhas que permitem o fluxo de energia (transmissão ou distribuição) no território brasileiro. Essas linhas que são armazenadas em dados do tipo *multilinestring*, conforme demonstra a Figura 12, criação da tabela. Outra tabela utilizada para analise, foi a que se refere ao conjunto de elementos agregados envolvendo trechos ferroviários, ou seja, ferroviárias cujo tipo seja "Ferrovia" e" Metrovia" com geometria do tipo linha.

```
CREATE TABLE enc_trecho_energia_l (
    id_objeto integer NOT NULL,
    numcircuitos integer,
    especie character varying(30),
    nome character varying(80),
    emduto character varying(3),
    tensaoeletrica double precision,
    geometriaaproximada character varying(3),
    operacional character varying(30),
    situacaofisica character varying(30),
    nomeabrev character varying(50),
    posicaorelativa character varying(30),
    geom geometry(MultiLineString,4674) NOT NULL
);
```

Figura 12 - Criação da tabela com tipo de dado Multilinestring
Fonte: Autoria própria.

4 ANÁLISE E COMPARAÇÕES

Com o resultado desse trabalho foi possível obter um comparativo de desempenho dos métodos de indexação no banco de dados PostgreSQL aplicados em dados geográficos. Ainda assim foi possível identificar melhores aplicações de cada um dos métodos aqui utilizados.

4.1 ANALISE DOS DADOS DO TIPO POINT

4.1.1 Inserções

As inserções foram feitas de três formas, primeiramente inserindo os dados no banco de dados sem aplicar nenhum método de indexação, resultado o qual é apresentado na Figura 13. Após essa análise, as inserções no BD foram realizadas depois da criação de índices nas tabelas vazias.

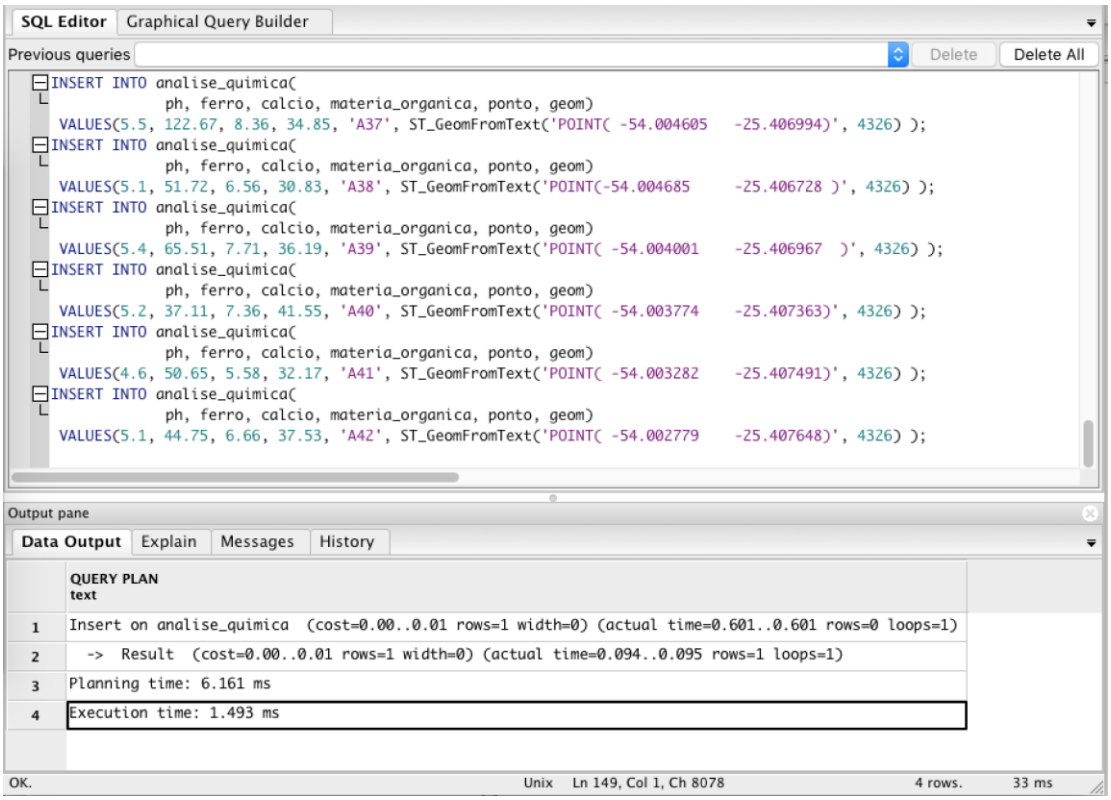


Figura 13 - Resultado da inserção dos dados sem indexação
Fonte: Autoria própria.

O primeiro tipo de índice criado na tabela de “analise_quimica” foi o GIST aplicado sobre a coluna “geom”, que indica a localização do ponto amostral, conforme o Quadro 8. O resultado nesse caso teve um desempenho insatisfatório, ou seja, o tempo de execução e o custo de processamento aumentou.

```
CREATE INDEX idx_geom_analise_quimica ON analise_quimica USING GIST(geom)
```

Quadro 8 - Criação índice GIST na tabela "analise_quimica"
Fonte: Autoria própria

A Figura 14, demonstra o resultado de inserção com índice do tipo GIST executado sobre o comando explain com a finalidade de verificar o custo dessa inserção.

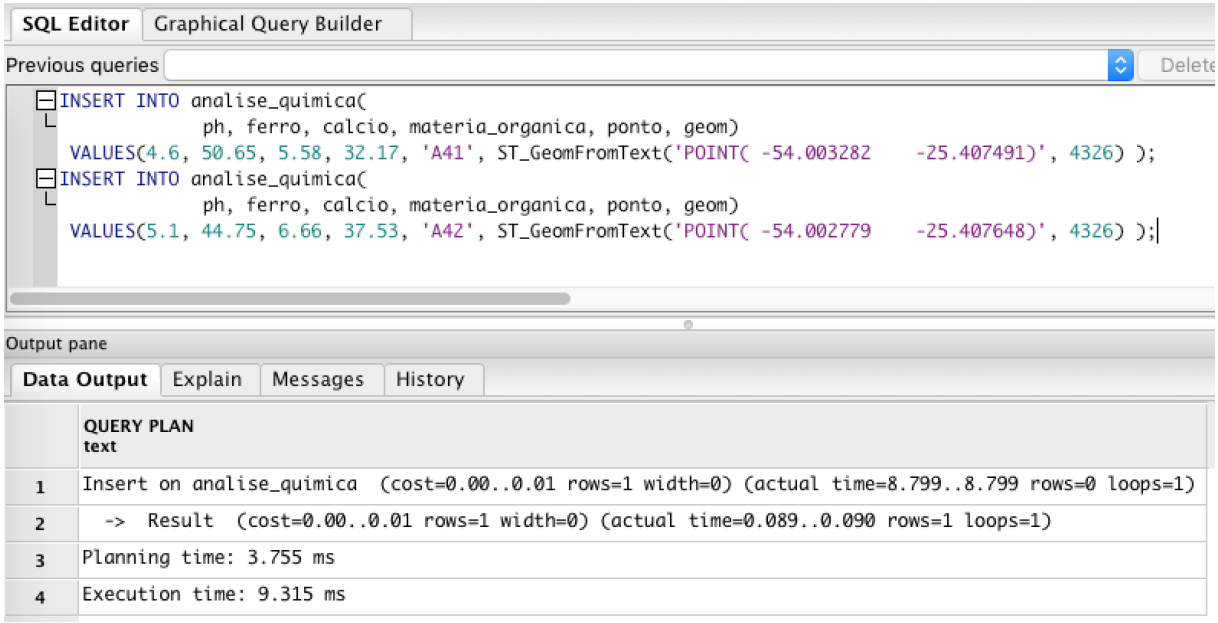


Figura 14 - Resultado de inserção com índice do tipo GIST
Fonte: Autoria própria

Executado o script de inserção apenas com o método de indexação do tipo B-Tree também criado sobre a coluna do dado espacial “geom”, conforme o Quadro 9, resultou em um ganho de desempenho comparado com as outras análises.

```
CREATE INDEX idx_geom_btree_analise_quimica ON analise_quimica (geom)
```

Quadro 9 - Criação índice B-Tree na tabela "analise_quimica"
Fonte: Autoria própria

A análise sobre o índice B-Tree representado na Figura 15.

	QUERY PLAN text
1	Insert on analise_quimica (cost=0.00..0.01 rows=1 width=0) (actual time=0.332..0.332 rows=0 loops=1)
2	-> Result (cost=0.00..0.01 rows=1 width=0) (actual time=0.009..0.009 rows=1 loops=1)
3	Planning time: 0.090 ms
4	Execution time: 0.586 ms

Figura 15 - Resultado de inserção com indexação B-Tree
Fonte: Autoria própria

Por fim, a análise dos dois índices criados representado na Figura 16.

	QUERY PLAN text
1	Insert on analise_quimica (cost=0.00..0.01 rows=1 width=0) (actual time=1.456..1.456 rows=0 loops=1)
2	-> Result (cost=0.00..0.01 rows=1 width=0) (actual time=0.255..0.256 rows=1 loops=1)
3	Planning time: 9.490 ms
4	Execution time: 1.940 ms

Figura 16 - Resultado de inserção com dois índices aplicados
Fonte: Autoria própria

Tabela 3 - Comparativo de resultados de desempenho na inserção dos dados

	Sem indexação	*B-Tree*	*GIST*	*B-Tree + GIST*
Tempo (em milissegundos)	1.493	0.586	9.135	1.940
Custo	0.601...0.601	0.332...0.322	8.799...8.799	1.456...1.456

Fonte: Autoria própria

A partir desses resultados, fica evidente o que foi anteriormente descrito, em que a utilização de métodos de indexação deve ser feita de maneira consciente, uma vez que o custo para a inserção dos dados com um índice do tipo GIST foi muito mais alta, em quase 900%.

Além disso, a aplicação de dois índices na mesma coluna não surtirá efeito se um dele tem um desempenho muito ruim. Dessa forma, fica explicito que deve ser estudado muito bem antes de utilizar de maneira erronia os índices.

4.1.2 Consultas

As consultas foram analisadas, através do comando *explain analyse* de quatro diferentes formas, a primeira sem a utilização de índice, a segunda aplicando índice do tipo B-Tree apenas no atributo “ponto”, a terceira aplicando o índice do tipo GIST nos atributos “geom”, que se tratam da localização do ponto e a quarta e última, fazendo uso dos dois índices descritos de uma só vez.

A criação dos índices para o fim de se obter o resultado da análise foi dividida em basicamente três processos, onde o índice era criado, executado a consulta com o operador *exeplain analyse*, para se obter o resultado de desempenho da consulta, posteriormente a tabela era apagada e criada novamente para que dessa forma a análise não sofresse interferência de dados já ordenados.

A realização da primeira análise se deu basicamente sobre uma consulta simples onde retorna todos dados da tabela sem nenhum critério, porém com uma coluna do tipo “geometry” que indica a localização do ponto amostral. Conforme mostra o Quadro 10.

```
SELECT * FROM analise_quimica;
```

Quadro 10 - Consulta simples para análise
Fonte: Autoria própria

A Figura 17 demonstra o resultado do desempenho da consulta sem nenhum método de indexação aplicado na tabela “analise_quimica”. É possível observar, na Figura 17, que a consulta teve um custo de execução inicial real no processador de 0.077 e final de 0.125, onde retornou 42 linhas (no caso é o total de linhas dessa tabela) em 0.181 milisegundos. O resultado do custo é calculado através de uma fórmula que multiplica a quantidade de páginas no disco que teve que ler por mil, somado com a quantidade de linhas multiplicado por 0.1, assim resultando o custo dessa consulta no processador.

	QUERY PLAN text
1	Seq Scan on analise_quimica (cost=0.00..2.84 rows=84 width=188) (actual time=0.077..0.125 rows=42 loops=1)
2	Planning time: 0.158 ms
3	Execution time: 0.181 ms

Figura 17 - Resultado consulta simples sem índice
Fonte: Autoria própria

Após essa análise, foi criado o índice do tipo B-Tree na coluna “ponto” da tabela, representado anteriormente no Quadro 8.

Já pode-se observar uma melhora considerável no desempenho da consulta, levando apenas 0.062 milissegundos e diminuindo também o custo para o processador. Conforme demonstra a Figura 18.

	QUERY PLAN text
1	Seq Scan on analise_quimica (cost=0.00..1.42 rows=42 width=188) (actual time=0.007..0.012 rows=42 loops=1)
2	Planning time: 0.274 ms
3	Execution time: 0.062 ms

Figura 18 - Resultado consulta simples com índice B-Tree
Fonte: Autoria própria

Após a realização dessa análise, a tabela “analise_quimica” foi totalmente excluída e inserida novamente, para que não sofresse interferência do índice B-Tree criado anteriormente. Dessa forma pode-se criar o índice do tipo GIST na coluna “geom” reordenando todos os dados novamente sobre esse método. Mesmo não diferindo muito do B-Tree, seus resultados foram consideráveis como demonstra a Figura 19.

```
CREATE INDEX idx_geom_analise_quimica ON analise_quimica USING GIST(geom)
```

Quadro 11 - Criação do índice do tipo GIST na tabela "analise_quimica"
Fonte: Autoria própria

	QUERY PLAN text
1	Seq Scan on analise_quimica (cost=0.00..1.42 rows=42 width=188) (actual time=0.008..0.012 rows=42 loops=1)
2	Planning time: 0.618 ms
3	Execution time: 0.063 ms

Figura 19 - Resultado consulta simples com índice GIST
Fonte: Autoria própria

E aplicado os dois índices na consulta o resultado obtido foi mais interessantes ainda, como pode-se observar na Figura 20.

	QUERY PLAN text
1	Seq Scan on analise_quimica (cost=0.00..1.42 rows=42 width=188) (actual time=0.008..0.014 rows=42 loops=1)
2	Planning time: 0.489 ms
3	Execution time: 0.041 ms

Figura 20 - Resultado consulta simples com dois índices
Fonte: Autoria própria

Tabela 4 - Comparativo de utilização de índice em uma consulta simples

	Sem indexação	*B-Tree*	*GIST*	*B-Tree + GIST*
Tempo (em milissegundos)	0.181	0.062	0.063	0.041
Custo	0.077...0.125	0.007...0.012	0.008...0.012	0.008...0.014

Fonte: Autoria própria

O resultado esperado da segunda consulta era encontrar os pontos amostrais da área em questão que se intersectam onde há uma produtividade menor a média e os dados da analise química também menor que a média, sendo possível analisar se há alguma relação entre a baixa produtividade do solo com suas baixas propriedades químicas.

Através de subconsultas, apresentada no Quadro 12, foi possível obter a média na própria consulta principal, fazendo assim com que ela se tornasse mais onerosa para o processador. Além disso, foi utilizado o operador topológico *St_Intersects* para encontrar os pontos das duas tabelas que satisfazem a regra imposta.

```
select p.ponto from produtividade p, analise_quimica a
       where p.produtividade < (select AVG(p1.produtividade) from produtividade p1)
               and a.ferro < (select AVG(a1.ferro) from analise_quimica a1)
               and a.ph < (select AVG(a1.ph) from analise_quimica a1)
               and a.calcio < (select AVG(a1.calcio) from analise_quimica a1)
               and a.materia_organica < (select AVG(a1.materia_organica) from analise_quimica a1)
               and St_Intersects(p.geom, a.geom)
```

Quadro 12 - Consulta com operador ST_Intersects
Fonte: Autoria própria

Seguindo os mesmos passos da primeira consulta, os índices foram aplicados na mesma ordem nas duas tabelas, “produtividade” e “analise_quimica”. A partir dos resultados pode-se confirmar a conclusão anteriormente tomada. Houve ganho de performance tanto no tempo de execução quanto no custo das consultas para o processador e leitura do disco, demonstrado na Tabela 5 e Figura 21.

Tabela 5 - Comparativo de índices B-Tree e GIST em consulta complexa

	Sem indexação	B-Tree	GIST	B-Tree + GIST
Tempo (em milissegundos)	2.778	1.086	0.839	0.793
Custo	1.471…2.125	0.491..0.873	0.298…0.721	0.286…0.677

Fonte: Autoria própria

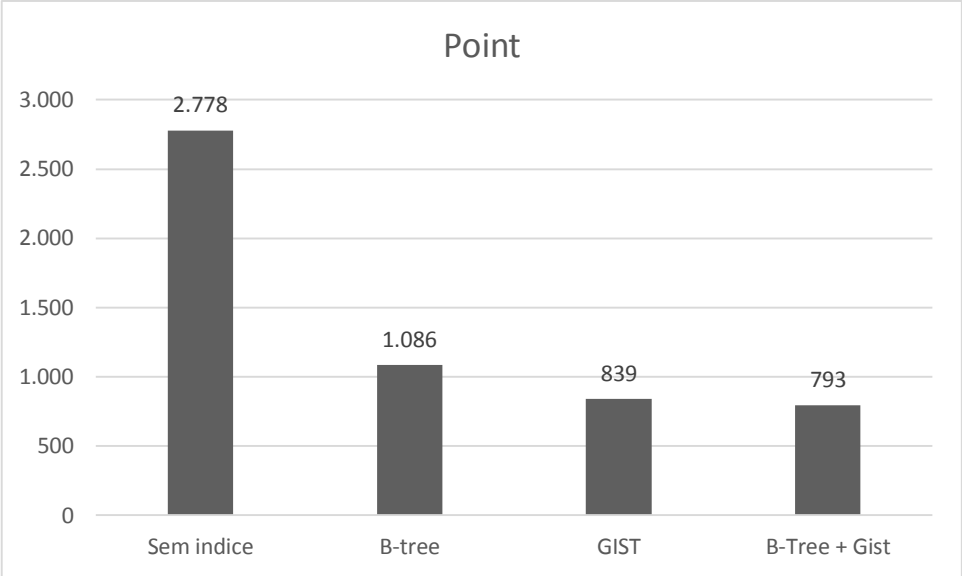


Figura 21 - Gráfico comparativo de desempenho aplicados à dados point
Fonte: Autoria própria.

4.2 ANALISE DOS DADOS DO TIPO MULTIPOINT, MULTIPOLYGON E MULTILINESTRING

Dada sobre uma consulta que se refere a encontrar pontos, linhas ou polígonos que se intersectam nas duas tabelas que tratam dados geográficos, a análise foi realizada seguindo os passos já descritos anteriormente. Ou seja, primeiramente analisando a consulta sem a inserção de índices, após com a inserção do índice B-Tree, excluída a tabela e adicionada novamente aplicando o índice Gist e para concluir mais uma análise feita com os dois índices aplicados nos dados *geometry*.

4.2.1 Resultado da consulta sobre dados *multipoint*

A análise realizada sobre dados do tipo *multipoint* foi dada através de uma consulta simples, mas que utiliza a função espacial *ST_Intersects*.

```
select * from asb_cemiterio_p a, eco_edif_industrial_p b
        where st_intersects(a.geom, b.geom);
```

Quadro 13 - Consulta sobre dados multipoint
Fonte: Autoria própria.

Tabela 6 - Resultado da consulta sobre dados multipoint

	Sem indexação	B-Tree	GIST	B-Tree + GIST
Tempo (em milissegundos)	1.892.909	0.790	317.509	0.071
Custo	0.00..644334.96	0.00..11330.62	0.14..1528.62	0.14..191.50

Fonte: Autoria própria.

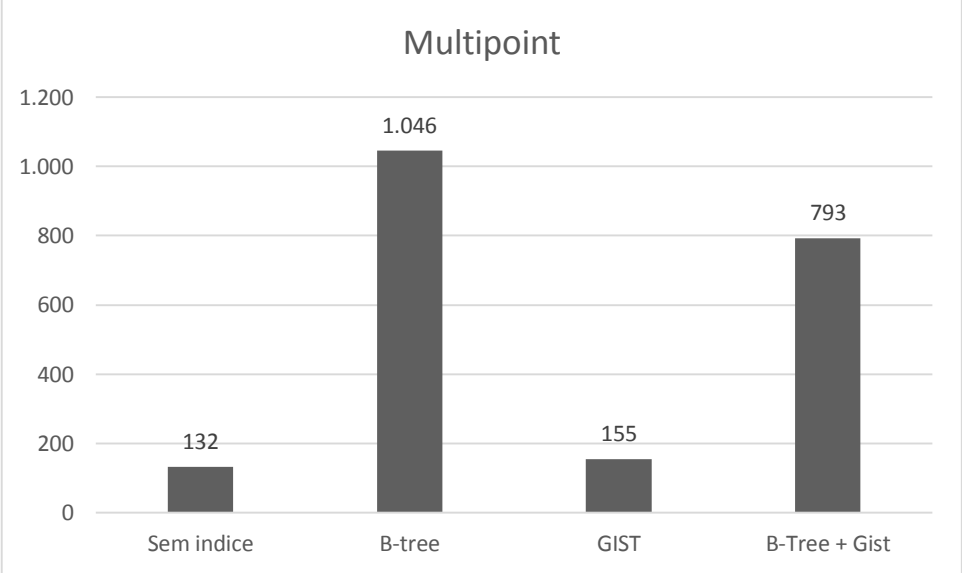


Figura 22 - Gráfico comparativo de desempenho aplicados à dados multipoint
Fonte: Autoria própria.

4.2.2 Resultado da consulta sobre dados *multipolygon*

No caso dos dados multipolygon, por ter dados muito grandes não foi possível aplicar o índice B-Tree. O PostgreSQL retornou o erro apresentado na Figura 23, dessa forma a análise foi realizada entre o retorno sem indexação e a indexação com o método GIST.

```
ERROR:  index row requires 66200 bytes, maximum size is 8191

********** Error **********

ERROR: index row requires 66200 bytes, maximum size is 8191
SQL state: 54000
```

Figura 23 - Erro criação índice B-Tree em dados Multipolygon
Fonte: Autoria própria.

```
select * from eco_area_agropec_ext_vegetal_pesca_a a, lim_unidade_federacao_a b
        where st_intersects(a.geom, b.geom)
```

Quadro 14 - Consulta sobre dados multipolygon
Fonte: Autoria própria.

Tabela 7 - Resultado da consulta sobre dados multipolygon

	Sem indexação	B-Tree	GIST	B-Tree + GIST
Tempo (em milissegundos)	535.108	0	524.220	0
Custo	0.00..20299.47	0	0.14..25.37	0

Fonte: Autoria própria.

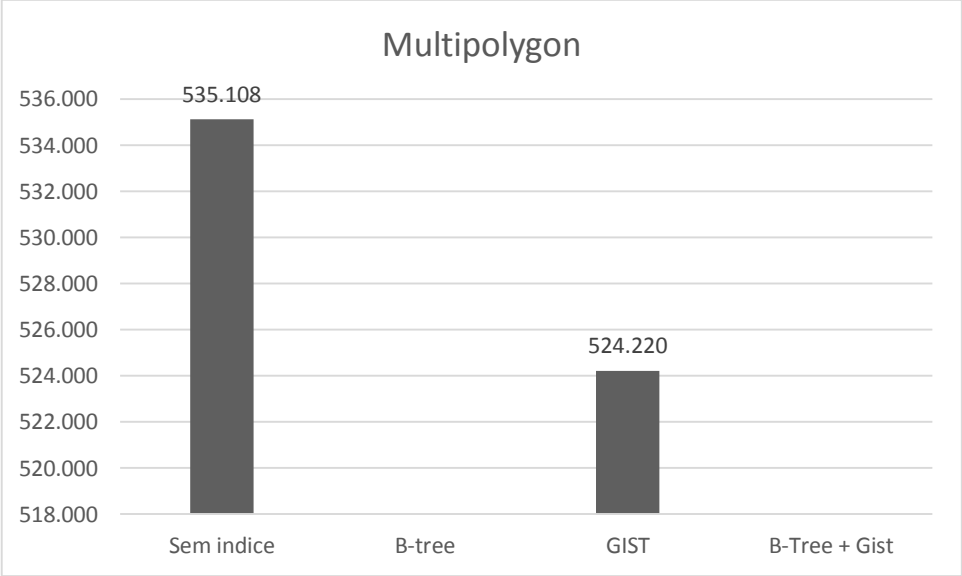


Figura 24 - Gráfico comparativo de desempenho aplicados à dados multipolygon
Fonte: Autoria própria.

4.2.3 Resultado da consulta sobre dados *multilinestring*

```
select * from enc_trecho_energia_l a, tra_trecho_ferroviario_l b
        where st_intersects(a.geom, b.geom)
```

Quadro 15 - Consulta sobre dados multilinestring
Fonte: Autoria própria.

Tabela 8 - Resultado da consulta sobre dados multilinestring

	Sem indexação	B-Tree	GIST	B-Tree + GIST
Tempo (em milissegundos)	1.892.909	0.790	317.509	0.071
Custo	0.00..644334.96	0.00..11330.62	0.14..1528.62	0.14..191.50

Fonte: Autoria própria.

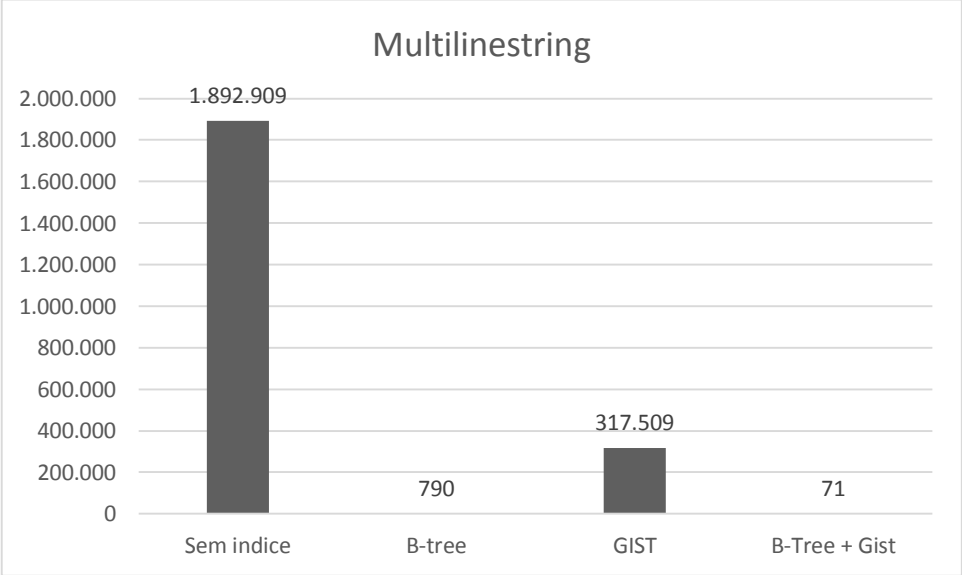


Figura 25 - Gráfico comparativo de desempenho aplicados à dados multilinestring
Fonte: Autoria própria.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÃO

A ferramenta PostgreSQL associado com a extensão PostGis oferece uma vasta área para a manipulação dos dados espaciais, tanto de armazenamento quanto de recuperação dos mesmos. Todavia, verificou-se que há ainda muitos problemas de desempenho ao se tratar de manipulação de dados espaciais, que se não analisados de forma criteriosa pode transformar o esforço para melhorar a performance em uma estrutura totalmente frágil e de difícil manutenção, que nem sempre alcançará o melhor desempenho.

Assim é que, a partir do estudo realizado no presente trabalho, a abordagem dos dados do tipo *point* o índice B-Tree se mostrou uma boa escolha, tanto na inserção quanto nas consultas.

O método de indexação do tipo GIST, avançado tecnologicamente, tanto que incorporou na versão 9.4 do PostgreSQL o índice do tipo R-Tree e que pode tratar de qualquer dado do tipo espacial nas consultas de forma afirmativa se apresenta mais performático do que o índice B-Tree independente do dado tipo de dado espacial tratado. Entretanto, quando o tipo de dado espacial é *Multilinestring*, o índice B-Tree é claramente mais performático que o Gist, diferente dos tipos *Multipoint* e *Multipolygon*.

Por fim, é possível concluir que há a necessidade de análises criteriosas para a utilização dos índices, a fim de que se encontre as melhores opções de quando e como utilizar essas metodologias.

5.2 SUGESTÃO PARA TRABALHOS FUTUROS

Como pode ser visto este trabalho deixou de apresentar comparativos com o tipo de dados raster, abordando apenas a espacialização de dados vetoriais, visto que, esta área de indexação é muito vasta e ampla. Além de realizar um estudo com demais tipos de métodos de indexação, logo o estudo sobre o tipo de dados raster seria grande valor, pois seria possível mapear possíveis comportamentos com todos os tipos de dados aplicando o tipo de índice B-Tree e Gist.

6 REFERÊNCIAS

ARONOFF, S. **Geographic Information Systems:** A Management Perspective. [S.l.]: Wdl Pubns, 1991.

BIANCHI, W. Entendendo e usando índices - Parte 1. **DevMedia**, 12 set. 2007. Disponivel em: <http://www.devmedia.com.br/entendendo-e-usando-indices-parte-1/6567>. Acesso em: 20 out. 2015.

BORGES, K. A.; DAVIS, C. F.; LAENDER, A. H. Modelagem conceitual de dados geográficos. **Divisão de Processamento de Imagens (DPI)**, 2005. Disponivel em: <http://www.dpi.inpe.br/livros/bdados/cap3.pdf>. Acesso em: 20 Abril 2015.

CÂMARA, G. **Representação computacional de dados geográficos**. Curitiba: MundoGEO, 2005.

CÂMERA, G. et al. CONCEITOS BÁSICOS EM GEOPROCESSAMENTO. **UFPA**, 07 Maio 2002. Disponivel em: <http://www.ufpa.br/sampaio/curso_de_sbd/sig/cap02-conceitos.pdf>. Acesso em: 20 Abril 2015.

CÂMERA, G.; CASANOVA, M. A. **Anatomia de Sistema de Informação Geográfica**. Campinas: UNICAMP, 1996.

CHAPTER 11. Indexes. **PostgreSQL 9.2.14 Documentation**, 1 Janeiro 2014. Disponivel em: <http://www.postgresql.org/docs/9.2/static/indexes-types.html>. Acesso em: 20 Outubro 2015.

COWEN, D. J. GIS versus CAD versus DBMS: What Are the Differences?. **Photogrammetric Engineering and Remote Snsing**, 1988.

DAVIS JR., C. A.; QUEIROZ, G. R. D. Métodos de acesso para dados espaciais. **Divisão de Processamento de Imagens (DPI)**, 2005. Disponivel em: <http://www.dpi.inpe.br/livros/bdados/cap6.pdf>. Acesso em: 21 Outubro 2015.

ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database System**. [S.l.]: Addison-Wesley, 2006.

FERREIRA, N. C. APOSTILA DE SISTEMA DE INFORMAÇÕES GEOGRÁFICAS. **UDESC**, 1 Janeiro 2006. Disponivel em: <http://www.faed.udesc.br/arquivos/id_submenu/1414/apostila_sig.pdf>. Acesso em: 20 Abril 2015.

FILHO, J. L. INTRODUÇÃO A SIG - SISTEMAS DE INFORMAÇÕES GEOGRÁFICAS. **Departamento de Informática - Centro de Ciências Exatas e Tecnológicas - UFV**, Dezembro 1995. Disponivel em: <http://www.dpi.ufv.br/~jugurta/papers/ti.pdf>. Acesso em: 20 Abril 2015.

FILHO, J. L.; IOCHPE, C. **INTRODUÇÃO A SIG – SISTEMAS DE INFORMAÇÕES GEOGRÁFICAS**. XVI Congresso da SBC. Recife: [s.n.]. 1996.

FRANK, A. U.; GOODCHILD, M. F. Two Perspectives on Geographical Data Modelling. **Techinal Paper**, Novembro 1990. 42.

GATRELL, A. C. Concepts of space and geographical data. **Longman Scientific &**, 1991. 119-134.

GOODCHILD, M. F. Geographical data modeling. **Computers &**, 1992.

GÜTING, R. H. An Introduction to Spatial Database Systems. **The VLDB Journal**, Berlin, v. 3, n. 4, p. 32, September 1994.

GUTTMAN, A. R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING, Berkeley, 06 Agosto 1984. Disponivel em: <http://www-db.deis.unibo.it/courses/SI-LS/papers/Gut84.pdf>. Acesso em: 21 out. 2015.

HELLERSTEIN, J. GiST: A Generalized Search Tree for Secondary Storage. **The GiST Indexing Project**, 06 Agosto 1999. Disponivel em: <http://gist.cs.berkeley.edu/>. Acesso em: 21 Outubro 2015.

KOTHURI, R. K. V.; RAVADA, S. Quadtree and R-tree Indexes in Oracle Spatial: A Comparison using GIS Data. **Department of Information and Computing Sciences**, 04 Junho 2002. Disponivel em: <http://www.cs.uu.nl/docs/vakken/dba/index_files/p546-kothuri.pdf>. Acesso em: 20 Outubro 2015.

LU, H.; OO, B.-C. Spatial Indexing: Past and Future. **Bulletin of the Technical Committee on Data Engineering**, Singapore, v. 16, n. 3, p. 6, 1993.

MAGUIRE, D. J.; LONGLEY, P. A. **Sistemas e Ciência da Informação Geográfica**. Porto Alegre: Bookman, v. 3, 2012.

NONATO, L. G.; VOORSLUYS, W. Quadtree. **Instituto de Ciências Matemáticas e de Computação Departamento de Computacao e Estatística SCE-5763 Tipos e Estrutura de Dados**, 25 Junho 2000. Disponivel em: <http://www.lcad.icmc.usp.br/~nonato/ED/Quadtree/quadtree.htm>. Acesso em: 21 Outubro 2015.

OLIVEIRA, H. P. D. ALTER INDEX. **Documentação do PostgreSQL**, 09 abr. 2007. Disponivel em: <http://pgdocptbr.sourceforge.net/pg82/sql-alterindex.html>. Acesso em: 20 out. 2015.

OLIVEIRA, H. P. D. CREATE INDEX. **Documentação do PostgreSQL**, 09 abr. 2007. Disponivel em: <http://pgdocptbr.sourceforge.net/pg82/sql-createindex.html#FTN.AEN48083>. Acesso em: 20 out. 2015.

OLIVEIRA, H. P. D. DROP INDEX. **Documentação do PostgreSQL**, 09 abr. 2007. Disponivel em: <http://pgdocptbr.sourceforge.net/pg82/sql-dropindex.html>. Acesso em: 20 out. 2015.

OLIVEIRA, H. P. D. EXPLAIN. **Documentação do PostgreSQL 8.2.0**, 09 Abril 2007. Disponivel em: <http://pgdocptbr.sourceforge.net/pg82/sql-explain.html>. Acesso em: 22 Outubro 2015.

PEDER, J. R. D.; SCHUCK, M. Estrutura de Índices para Arquivos. **Unioeste**, Cascavel, 20 set. 2009. Disponivel em: <http://www.inf.unioeste.br/~olguin/4458-semin/G7-monografia.pdf>. Acesso em: 20 out. 2015.

POSTGIS. Chapter 4. Using PostGIS: Data Management and Queries. **PostGIS**, 2015. Disponivel em: <http://postgis.net/docs/manual-2.1/using_postgis_dbmanagement.html#PostGIS_Geography>. Acesso em: 20 Outubro 2015.

POSTGIS. Documentation PostGIS 2.2. **PostGIS**, 2015. Disponivel em: <http://postgis.net/stuff/postgis-2.2.pdf>. Acesso em: 21 Outubro 2015.

POSTGRESQL. Indexes. **Documentação do PostgreSQL**, 10 out. 2015. Disponivel em: <http://pgdocptbr.sourceforge.net/pg80/indexes.html>. Acesso em: 20 out. 2015.

POSTGRESQL. PostgreSQL About. **PostgreSQL**, 2015. Disponivel em: <http://www.postgresql.org/about/>. Acesso em: 20 Outubro 2015.

POSTGRESQL, D. Chapter 56. GiST Indexes. **PostgreSQL 9.4.5 Documentation**, 5 Agosto 2014. Disponivel em: <http://www.postgresql.org/docs/9.4/static/gist-extensibility.html>. Acesso em: 21 Outubro 2015.

QUADTREE. **Wikipedia**, 05 Outubro 2015. Disponivel em: <https://en.wikipedia.org/wiki/Quadtree#/media/File:Quad_tree_bitmap.svg>. Acesso em: 20 Outubro 2015.

RICARDO, L. R.; PIRES, L. B. L. C. Otimização de consultas no PostgreSQL. **Revista SQL Magazine**, n. 39, 2013.

SANTANA, G. Entendendo e balanceando índices Btree. **Comunidade Firebird de Língua Portuguesa**, 2012.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de bancos de dados**. [S.l.]: Makron Books, 1999. 778 p.

SMITH, T. et al. KBGIS-II: a knowledge-based geographic information system.. **International Journal of Geographic Information Systems,** 1987.

TOMA, L. I. Laura I. Toma. **Bowdoin**, 01 Maio 2008. Disponivel em: <http://www.bowdoin.edu/~ltoma/teaching/cs340/spring08/Papers/Rtree-chap1.pdf>. Acesso em: 21 Outubro 2015.

TYPES, I. PostgreSQL 8.1.23 Documentation. **PostgreSQL**, 26 Setembro 2011. Disponivel em: <http://www.postgresql.org/docs/8.1/static/indexes-types.html>. Acesso em: 21 Outubro 2015.

WINAND, M. The Search Tree (B-Tree) Makes the Index Fast. **Use the Index Luke**, 13 Outubro 2011. Disponivel em: <http://use-the-index-luke.com/sql/anatomy/the-tree>. Acesso em: 20 Outubro 2015.

YE, Y. More trees. **Mendel Informatics Indiana**, 2014. Disponivel em: <http://mendel.informatics.indiana.edu/~yye/lab/teaching/spring2014-C343/moretrees.php>. Acesso em: 20 Outubro 2015.